WEST Search History

DATE: Monday, September 29, 2003

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L17	L16 and l6	6	L17
L16	L15 same l2 same l3	8	L16
L15	initial near5 seed	1824	L15
L14	intial near5 seed	1	L14
L13	L12 and l6	7	L13
L12	L11 same l9	40	L12
L11	encrypt$	15122	L11
L10	L9 and l6	46	L10
L9	L8 same l1	96	L9
L8	key	278571	L8
L7	L6 and l5	5	L7
L6	@ad<19970715	2510314	L6
L5	L4 and l1	11	L5
L4	L3 same l2	97	L4
L3	new near7 seed	3425	L3
L2	random$ near7 number	24225	L2
L1	2048 near5 bit	2216	L1

END OF SEARCH HISTORY

L13: Entry 1 of 7 File: USPT Feb 22, 2000

DOCUMENT-IDENTIFIER: US 6028933 A
TITLE: Encrypting method and apparatus enabling multiple access for multiple services and multiple transmission modes over a broadband communication network

Application Filing Date (1):
19970417

Detailed Description Text (119):
FIG. 26 is an exemplary representation of the use of a Rabin's method certified public key with a cable modem public component to yield mutual authentication between headend and cable modem, in accordance with the present invention. If the key exchange and mutual authentication are to be completed at Level 2, then step 2465 of FIG. 24 directs a transition to step 2605 of FIG. 26. In step 2605, the HE transmits Certificate A to the cable modem. Certificate A is mathematically represented as: sqrt(SN,n.sub.OS) mod n.sub.KCA and is concatenated with the KCA's public key (n.sub.KCA) within the message that transmits the certificate. The certificate contains the cable modem's SN and the OS' public key (n.sub.OS), encrypted modulo n.sub.KCA. In step 2610, the CM performs a hash function on the received n.sub.KCA. The CM then compares the hash function result with the hash of n.sub.KCA that is stored in NVM as programmed by the manufacturer, in accordance with step 2615. If the result of the comparison is that the hash functions do not match, then the certification process is begun anew at step 2605. If the hash function results do match, then the CM accepts the full 2048 bit n.sub.KCA as authentic, per step 2620.

Detailed Description Text (125):
FIG. 28 is the completion of an exemplary representation of a Level 2 process of mutual authentication between headend and cable modem, begun in accordance with FIG. 26, continued in accordance with FIG. 27, and in accordance with the present invention. In step 2805, the PMK is written to the NVM at the HE adjacent to PMK#.sub.CM. PMK#.sub.CM is then incremented in NVM, in accordance with step 2810. In step 2815, IV.sub.AUTH is encrypted with RSK to form an authenticated shared secret key (SSK.sub.AUTH). The CM next formulates a Rabin encrypted message to send to the HE containing the CM serial number, PMK, IV.sub.PMK, and SSK.sub.AUTH, in accordance with step 2820. A random number is provided for padding (PAD) to 2048 bits. When using Rabin' method to either encrypt or decrypt, it is necessary to provide random padding to the data to be encrypted or decrypted so as not to denegrate the level of security, as would be apparent to those skilled in the art. The terms are concatenated and Rabin encrypted by squaring the concatenation modulo n.sub.headend. The resulting encrypted message is transmitted from the CM to the HE, and is mathematically represented as: (SN, PMK, IV.sub.PMK, SSK.sub.AUTH, PAD).sup.2 mod n.sub.headend, in accordance with step 2825. In step 2830, the HE decrypts the message transmitted in step 2825 by taking the square root modulo n.sub.headend of the message. The HE then compares the decrypted IV.sub.PMK to the IV.sub.PMK stored in memory, in accordance with step 2835. If the result of step 2835 is negative, then the HE directs that the Level 2 key exchange and authentication process begins anew at step 2605 of FIG. 26. If, however, the IV.sub.PMK comparison yields a match, then the HE decrypts E.sub.SSKauth in accordance with step 2840. The decrypted value of the cable modem SN is then compared to the SN stored at the HE, in accordance with step 2845. If the result of step 2845 is negative, then the HE directs that the Level 2 key exchange and authentication process begins again at step 2605 of FIG. 26. If the result of step 2845 is affirmative, however, then the HE writes PMK and PMK#.sub.HE-RAM into NVM, in accordance with step 2850. The CM and HE have completed mutual authentication and now both possess a mutually created and mutually authenticated permanent key (PMK). The

PMK is designed to be unbreakably encrypted for twenty years, and as such, the public key exchange described is intended to occur only once in the twenty year life of the cable modem, although if the PMK was compromised, a new public key exchange could be accomplished, resulting a new PMK. Advantageously, since the present invention doesn't require a public keying transaction at each connection or power up, delay time is avoided at each connection or power up. Once the PMK is generated, unavailability of the KCA or OS does not affect the present invention at the time of subsequent connections or power up.

WEST

Generate Collection | Print

L13: Entry 2 of 7 File: USPT Oct 19, 1999

DOCUMENT-IDENTIFIER: US 5970144 A
TITLE: Secure authentication-key management system and method for mobile communications

Application Filing Date (1):
19970131

Detailed Description Text (14):
The validator A-key generation/distribution unit 514 identifies 706 the ESN of the MS 102 by reading the information from the MS identification unit 610. The validator A-key generation/distribution unit 514 then encrypts the ESN of the MS 102 using the RC4 encryption algorithm. The RC4 encryption algorithm is described in greater detail in K. R. Stamberger, "The RC2 and RC4 Exportable Encryption Algorithms," RSA Data Security, Inc. (Feb. 12, 1993). In general, the RC4 algorithm is a symmetric stream encryption algorithm. A stream cipher processes the input data a unit at a time. A unit of data is generally a byte, or bit. In this way, encryption or decryption can execute on a variable length of input. The algorithm does not have to wait for a specified amount of data to be input before processing, or append and encrypt extra bytes. RC4 is actually a keyed pseudo-random sequence. It uses the provided key to produce a pseudo-random number sequence which is logically combined using an exclusive-OR (XOR) operation with the input data. As a result the encryption and decryption operations are identical. The number of key bits is variable and can range from eight to 2048 bits.

WEST

Generate Collection Print

L13: Entry 3 of 7 File: USPT Dec 15, 1998

DOCUMENT-IDENTIFIER: US 5850445 A
TITLE: Authentication key management system and method

Application Filing Date (1):
19970131

Detailed Description Text (14):
The validator A-key generation/distribution unit 514 identifies 706 the ESN of the MS 102 by reading the information from the MS identification unit 610. The validator A-key generation/distribution unit 514 then encrypts the ESN of the MS 102 using the RC4 encryption algorithm. The RC4 encryption algorithm is described in greater detail in K. R. Stamberger, "The RC2 and RC4 Exportable Encryption Algorithms," RSA Data Security, Inc. (Feb. 12, 1993). In general, the RC4 algorithm is a symmetric stream encryption algorithm. A stream cipher processes the input data a unit at a time. A unit of data is generally a byte, or bit. In this way, encryption or decryption can execute on a variable length of input. The algorithm does not have to wait for a specified amount of data to be input before processing, or append and encrypt extra bytes. RC4 is actually a keyed pseudo-random sequence. It uses the provided key to produce a pseudo-random number sequence which is logically combined using an exclusive-OR (XOR) operation with the input data. As a result the encryption and decryption operations are identical. The number of key bits is variable and can range from eight to 2048 bits.

WEST

Generate Collection | Print

L13: Entry 4 of 7 File: USPT Jun 16, 1998

DOCUMENT-IDENTIFIER: US 5768373 A
TITLE: Method for providing a secure non-reusable one-time password

Application Filing Date (1):
19960506

Detailed Description Text (11):
Generally, encryption/decryption units 218/224 operate in accordance with an asymmetric algorithm such as those found in RSA Corp.'s Bsafe Library.TM., a publicly available product. Further, the public-private key pair generated by public-private key generator 248 may generally be between 360 and 2048 bits in length. However, use of at least 768 bits is preferred because use of any fewer bits may result in easily penetrable security.

WEST

Generate Collection Print

L13: Entry 5 of 7 File: USPT Mar 31, 1992

DOCUMENT-IDENTIFIER: US 5101432 A
**** See image for Certificate of Correction ****
TITLE: Signal encryption

Application Filing Date (1):
19900116

Detailed Description Text (20):
In a preferred embodiment, the keys to the encryptor and decryptor each consist of 2048 16-bit words while the encryptor input data consists of 13-bit words. Shorter keys may be implemented through software modifications. Short lengths are desirable for increased speed of the system, and in many applications the apparatus hardware may be connected for short lengths only.

WEST

Generate Collection Print

L13: Entry 6 of 7 File: USPT Apr 16, 1991

DOCUMENT-IDENTIFIER: US 5008935 A
TITLE: Efficient method for encrypting superblocks of data

Application Filing Date (1):
19890630

Brief Summary Text (6):
In accordance with the principles of this invention, a superblock or buffer of data comprising a plurality of blocks, each block containing an integral number of words, is encrypted in the following way. First, each block is encrypted using a block encryption arrangement (such as a DES encryption or a simpler program-controlled encryption) and a first key. The outputs of these encryptors are organized into words and these output words from all of the encryptors are permuted according to a second key to generate the encrypted output. Advantageously, the second encryption key can have N! values which will make it very difficult to attack an encrypted message when N is a number such as 256 representing a buffer of 256 words (bytes) or 2,048 bits.

WEST

End of Result Set

☐ Generate Collection Print

L13: Entry 7 of 7 File: USPT Feb 3, 1981

DOCUMENT-IDENTIFIER: US 4249180 A
TITLE: Past dependent microcomputer cipher apparatus

Application Filing Date (1):
19780920

Detailed Description Text (32):
Two basic levels of cipher keys are provided by and are stored in the apparatus 10. An electrically reprogrammable PROM 37 stores a random 2048 bit pattern and is used as a cipher key on a `corporate` level. Each apparatus pair 10 and 10' within a system as in FIG. 2 would therefore carry the same random pattern of bits and could readily communicate in the encryption format. For the user who requires a higher level of security above the corporate level within the same system, it is possible to program an individual key into an apparatus 10. The operator merely presses an ENTER KEY pushbutton at a keyboard (not shown) of the terminal 12 and proceeds to type characters from the keyboard until an ENTER KEY indicator (not shown) of the apparatus 10 extinguishes. The number of characters required is fifteen and all further characters are ignored. The characters input can be alphabetical or numerical in any combination and are used to transpose the 2048 bit pattern in the PROM 37 into a new and individual pattern in the RAM 33. Most users would, however, probably employ an easily remembered phrase or number combination for developing individual cipher keys. The apparatus 10 is retired from the key entry mode by again depressing the ENTER KEY pushbutton. By this method a requirement for a large family of individual keys may be obtained and easy changing of keys is readily implemented. Return to the corporate key from the the individual key is accomplished by operating the reset switch 29.

WEST

Generate Collection | Print

L17: Entry 1 of 6 File: USPT Sep 26, 2000

DOCUMENT-IDENTIFIER: US 6125186 A
TITLE: Encryption communication system using an agent and a storage medium for storing that agent

Application Filing Date (1):
19970708

Detailed Description Text (95):
The initial seed is set in step S11. The method of setting the initial seed is as was explained with reference to FIG. 11. In step S12, pseudo random numbers are generated using the pseudo random number generator based on that initial seed. In steps S13 and S14, the generated random numbers are sent to the seed section as appropriate seeds. In step S15, the timing at which the seed is changed is determined. This change timing is shown by, for example, a parameter such as number of packets or time. In step S16, whether or not the time has reached the seed changing timing is monitored. When the timing to change the seed is reached, in step S17 the immediately preceding generated pseudo random number is set as the new seed, and the procedure returns to step S12. After that, steps S12 to S17 are repeated. The seed is changed at irregular intervals according to the processing described above.

WEST

Generate Collection | Print

L17: Entry 2 of 6 File: USPT May 18, 1999

DOCUMENT-IDENTIFIER: US 5905445 A
TITLE: Keyless entry system with fast program mode

Application Filing Date (1):
19970505

Brief Summary Text (9):
During use, a normal command is generated by pressing one transmitter button and a message is transmitted in the form shown in FIG. 4 including a preamble, a function code, the transmitter ID, a sequence number, an authenticator, and a CRC. If the transmitter normal message is sent a few times when the receiver is out of range, the receiver loses synchronization with the transmitter but can catch up by using the sequence number to resynchronize. If the receiver lags in sequence by a given amount such as 264 sequence numbers, it cannot automatically resynchronize. Then it is necessary to transmit a Resynch command which is like the normal command of FIG. 4 except that a randomly selected sequence number is used. When the Resynch command is given, the initial seed is used along with the new sequence number to determine the authenticator in both the transmitter and the receiver.

WEST

Generate Collection | Print

L17: Entry 3 of 6 File: USPT Mar 24, 1998

DOCUMENT-IDENTIFIER: US 5732138 A
TITLE: Method for seeding a pseudo-random number generator with a cryptographic hash of a digitization of a chaotic system

Application Filing Date (1):
19960129

Detailed Description Text (4):
FIG. 1 is a flowchart describing the steps in creating a sequence of random numbers. An overall description of these steps is given in this paragraph. A more detailed description of these steps will be elaborated on in the rest of the description of the invention with reference to the currently preferred implementation. In the first step 100, the state of a chaotic system is recorded. The state is then converted into a sequence of bits to form a binary string, step 105. This sequence of bits is then modified using a cryptographic hash function, step 110. And from this modified sequence of bits one obtains yet another sequence of bits, which is designated as the "seed," step 115. This seed is then used as the initial state for a pseudo-random number generator, step 120. The pseudo-random number generator continues to produce numbers as desired, step 125. A new seed can be generated at any time by repeating the process by repeating step 120. It should be noted that, as an option, the pseudo-random number generator could be removed from the system. In which case, a new seed would be generated from the chaotic source each time a new random number is needed.

Detailed Description Text (18):
The function of the pseudo-random number generator in regard to the current invention is described in FIG. 6. Initially, the pseudo-random number generator receives a seed, step 600. For the currently preferred embodiment, the seed is obtained by applying a cryptographic hash function to the output from the chaotic source and taking the appropriate number of bytes. The seed is then used as the initial state, step 605. In step 610, the pseudo-random number generator repeatedly takes this state and applies a deterministic transformation to obtain a new state and a random number. This process is repeated as desired, steps 610 and 615. As the pattern generated will eventually cycle, one may wish to periodically refresh the system by obtaining a new seed. The time period between refreshes of the seed may be determined by the cryptographic application.

WEST

Generate Collection | Print

L17: Entry 5 of 6 File: USPT Jan 17, 1995

DOCUMENT-IDENTIFIER: US 5383143 A
TITLE: Self re-seeding linear feedback shift register (LFSR) data processing system for generating a pseudo-random test bit stream and method of operation

Application Filing Date (1):
19940330

Detailed Description Text (4):
The current invention uses a linear feedback shift register (LFSR) to create pseudo-random patterns from an initial seed (starting state). Also, the novel LFSR described herein uses the tact that an LFSR, having N bits of storage, cycles through all possible [(2.sup.N) -1] binary states, except the all zero state, to create it's own next and different seed value. Therefore, the LFSR provides [(2.sup.N)-1] values for one seed (i.e., the first seed). The LFSR is designed herein to choose one of these [(2.sup.N)-1] values as a next new seed which is numerically different from the first seed in order to generate ([(2.sup.N)-1]) more pseudo-random numbers in addition to the [(2.sup.N)-1] pseudo-random numbers generated via the first seed. Therefore, the LFSR taught herein can generate approximately ([(2.sup.N)-1])*([(2.sup.N)-1]).apprxeq.2.sup.2N unique pseudo-random values via [(2.sup.N)-1] seeds by re-seeding itself (i.e., replacing an old seed with a new seed wherein the new seed is selected from a pseudo-random number generated by the LFSR when using the old seed).

Detailed Description Text (5):
Specifically, the next/new seed value is installed by allowing a second register, that is of equal length to the LFSR, to capture a state (i.e., a pseudo-random number) from the LFSR that is not equivalent to the previous/old seed. This value is used as a compare value to indicate when the next seed value must be captured and when the scan chain control signals must be applied. This is significantly different than the common method of storing a set of seeds (initial states) or polynomial values (feedback connections) in N RAM or ROM memory elements and providing means to access and apply these stored values. It is also different from the other common methods that rely on creating seed values or polynomial values by using counters. The method of this invention uses far fewer logical circuit elements in it's implementation of overall data generation than either the RAM/ROM storage method or the counter method and consequently reduces the amount of circuitry needed.

WEST

End of Result Set

☐ Generate Collection Print

L17: Entry 6 of 6 File: USPT Nov 2, 1993

DOCUMENT-IDENTIFIER: US 5258936 A
TITLE: Method and apparatus for generating pseudo-random numbers

Abstract Text (1):
A method and apparatus for generating pseudo-random numbers. A programmably selectable MASK value determines the polynomial to be used to generate the pseudo-random numbers. The MASK value can be changed while pseudo-random numbers are being generated in order to increase the run rate and improve the randomness of the sequence of pseudo-random numbers being generated. A programmably selectable SEED value is also used. The initial SEED value is used to generate the first pseudo-random number. The first pseudo-random number is then used as the NEW SEED value to generate the second pseudo-random number, and so on.

Application Filing Date (1):
19930510

Detailed Description Text (19):
Using bus 28, a user can load a desired SEED value into seed register 14. Once a user programs an initial SEED value into seed register 14, the SEED value stored in seed register 14 is continuously updated by the pseudo-random number generation procedure which is described in TABLE 1 above. The user may, but need not, change the SEED value in the seed register 14 by loading in a new value across bus 28.

Detailed Description Text (35):
The present invention also allows the user to select the initial SEED value under program control. The initial SEED value is used to generate the first pseudo-random number. The first pseudo-random number is then used as the NEW SEED value to generate the second pseudo-random number, and so on. Although the procedure for generating pseudo-random numbers does not require the user to change either the MASK value or the SEED value, the user is free to change either or both the MASK value or the SEED value while pseudo-random numbers are being generated.